

Developing Data Intensive Web Sites

*A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology*

by
M. Subbarayudu

to the
**Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur
March, 1998**

23 APR 1998

CENTRAL LIBRARY
KAMPUC

Inv. No. A 125379

CSE-1998-M-SUB-DEV

Entered in system

12810
23698



A125379

Certificate

Certified that the work contained in the thesis entitled "*Developing Data Intensive Web Sites*", by Mr.M. Subbarayudu, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.



(Dr. T.V.Prabhakar)

Professor,

Department of Computer Science & Engineering,

Indian Institute of Technology,

Kanpur.

March, 1998

Acknowledgments

I am always grateful to my thesis supervisor, **Dr. T. V. Prabhakar** for giving me an interesting problem. I want to thank him for his support and guidance throughout my work. In spite of being busy with various appointments he was always available and patient.

I would also like to thank my friends K. V. Vihari, Gorti Sriram, Sandeep and others who had helped me a lot by giving new ideas. I am thankful to all of my friends and my batch-mates who made my IITK life interesting, inspiring and memorable one. I wish to thank my family for encouraging to do my M.Tech and for tolerating my absence from their midst.

Abstract

Nowadays there is a great amount of interest in building hypertext/hypermedia systems that is leading to the discovery of new ways of organizing information. In this thesis we have proposed a methodology for the design and development of hypertext interfaces to existing databases. We also developed a design tool that helps in the design and construction of such interfaces according to the steps of our methodology.

Our methodology is organized into a sequence of seven steps: Conceptual modeling, Physical mapping, Navigational modeling, Conversion into SNS templates and HTML templates, User interface modeling, Functional modeling and Implementation. Starting with the first step the designer determines the structure of the hyperbase, its physical mapping with database, navigational possibilities, interface specifications and the functionalities to be incorporated into the application. In the implementation the designer has to construct a web interface according to the design specifications determined in the first six steps and has to write a presentation system that accepts requests and presents data demanded by the user.

The Web Interface for DataBase(WIDB) Design tool we developed automates the steps of our methodology in the construction of websites that act as an interface to browse through an existing database.

Contents

1	Introduction	1
1.1	Introduction	1
1.2	Background	3
1.3	Typical website goals	4
1.4	Motivation	4
1.5	Thesis organization	6
2	Related work	7
2.1	Hypertext Data Modeling(HDM)	7
2.2	Relational Management Methodology(RMM)	8
2.3	Systematic Hypermedia Application Design with OOHDM	10
2.4	Browsing A Database	12
3	Methodology for Design of Data intensive Web Sites	14
3.1	Data Model	14
3.1.1	Structural Primitives	14
3.1.2	Access Structures	16
3.2	Methodology	17
3.2.1	Conceptual modeling	17

3.2.2	Physical Mapping	20
3.2.3	Navigational Modeling	20
3.2.4	Conversion into SNS templates	23
3.2.5	Interface Modeling	24
3.2.6	Functional Modeling	25
3.2.7	Implementation	26
4	Web Interface for DataBase(WIDB) Design Tool	27
4.1	Conceptual Modeling	28
4.2	Navigational Modeling	30
4.3	Conversion into SNS Templates	31
4.4	User Interface	32
4.5	The Save Option	33
5	IMPLEMENTATION	34
5.1	Implementation of WIDB Tool	34
5.2	Implementation of Presentation System	38
6	Conclusions	41
6.1	Summary	41
6.2	Further Extensions	43

Chapter 1

Introduction

1.1 Introduction

The increasing importance of the World Wide Web(WWW) has spurred the creation of large number of websites. Increasingly large amounts of information is made available for the users of the WWW. With this has come, recently a greater amount of interest to build information systems that are accessible through the WWW. One of the outcomes of the popularity of the WWW is the wide spread use of the concept of hypertext.

Hypertext is a mechanism which provides a novel and non_sequential method of accessing information unlike traditional systems which are sequential in nature. Hypertext has been defined as "an approach to information management, in which data is stored as a network of nodes, connected together by links" [2]. A node usually represents a single concept or idea and its elements can be text, graphics, images, audio, video, animation etc. Words, phrases or an image within a node can be associated with other nodes containing related information. For this the designer has to create a link between the word, phrase or image and the related information [7]. As multimedia technology comes age, the concept of hypertext is extended to the more general concept hypermedia.

An illustration of hypertext system is shown in figure-1. The browser in the top of figure illustrates that the node A has been selected for the display of its contents and it shows the links to nodes B,G to which it was connected in the hypertext database [2].

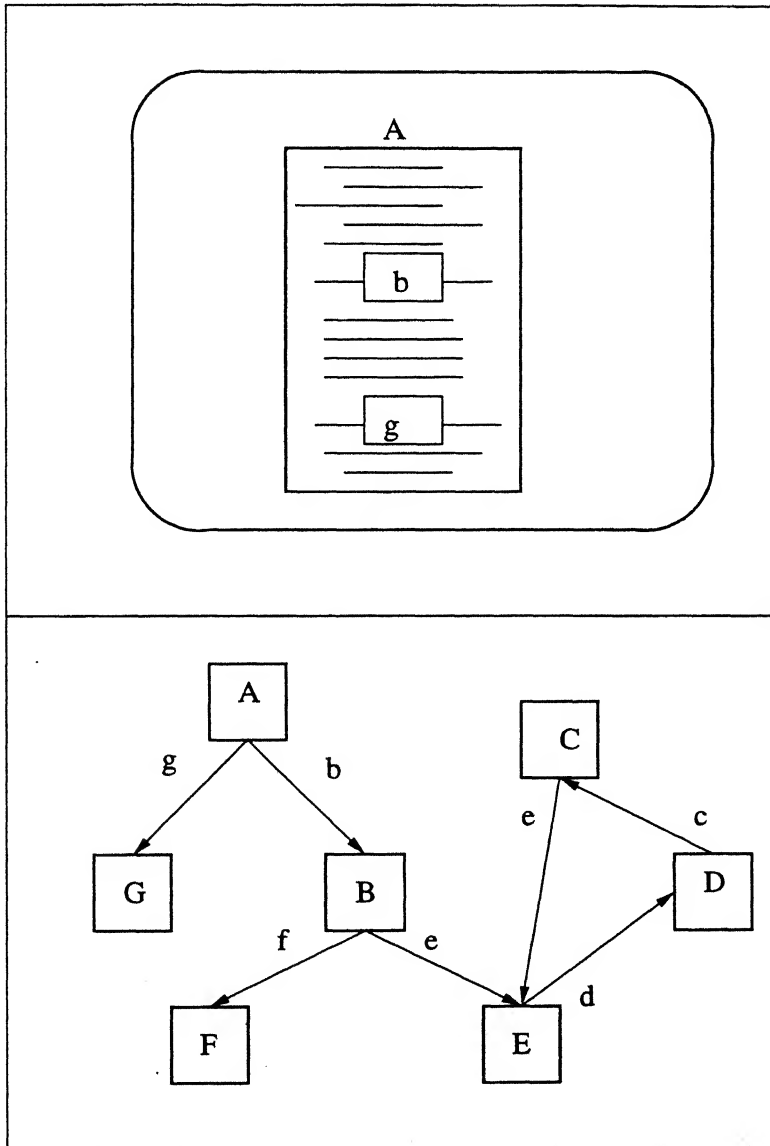


Figure 1: Illustration of navigation through Hypertext database

1.2 Background

The following concept definitions of Hypertext/Hypermedia Systems are necessary [7] [2] [1].

- **Anchor**:- The starting point of a link is known as an anchor point. A user can click on an anchor and the associated link will be traversed, taking the user to the associated node.
- **Navigation**:- The process of leaping from one node to its linked node by clicking on an anchor is navigation.

There are two methods for explicitly linking two points in hypertext systems.

- **Organizational links**:- These links are used to implement hierarchical information. Organizational links connect a parent node with its children and thus form a strict tree subgraph within the hypertext network graph. Organizational links are often traversed by a separate mechanism at the node control level(i.e goto-parent, goto-first-child, goto-next-sibling etc).
- **Referential links**:- These links can be thought of as the linking method that distinguishes hypertext and make it successful. These are used to link non_hierarchical information. They represent the domain dependent associations among the nodes in the hypertext system. They generally have two ends and are usually directed.
- **Typed nodes**:- While determining the nodes of a hypertext/hypermedia application all possible nodes in the domain of application can be sorted into different types or classes. Then each class of nodes can be represented as a typed node in the design of the application. The associations between typed nodes are usually represented as link types.
- **HTML Templates**:- These are semistructured - typed nodes which contain labeled fields formatted properly and spaces for field values. Instantiation of a

HTML template yields an empty document in which the user can enter data for various fields of the document.

- **Structured nodes:-** These are essentially typed nodes having some internal structure. If a hypermedia application is designed using structured nodes then there can be a possibility of some processing on this hyper database similar to traditional databases. Moreover this will help to build tools that can automate the design and implementation of hypermedia applications. Recently many methodologies have been proposed for the structured design of hypermedia applications, but they do not support the use of structured nodes in the conversion and implementation parts.

1.3 Typical website goals

A primary way of categorizing websites is by goals of the originators, as interpreted by the designers. The website goals tied to typical organizations are shown in Table-1[13].

1.4 Motivation

Hypertext and hypermedia systems are used in many applications because of their flexible structure and the great browsing freedom they give to users. But building hypertext/hypermedia applications on large scale is not an easy task. So in the recent past some methodologies for systematic design of hypermedia applications have been proposed among which RMM [14] and OOHDM [4] are the well known ones. Such methods are best suited for designing front ends to loosely structured data(as pointed out in RMM). Moreover their main focus is to design only those hypermedia applications that are a collection of static nodes interconnected through static links.

But many organizations(some are listed in Table-1) are competing to bring their large databases to the potential customers/users accessible through internet

Purpose	Some Organizations
Sell products	Publishers, airlines, departmental stores
Advertise products	NBC, Ford, IBM, Microsoft, Sony
Inform and announce	Universities, museums, cities
Provide access	Libraries, news papers, scientific organizations
Offer services	Governments, public utilities
Credit discussions	Public interest groups, magazines
Nurture communities	Political groups, professional associations

Table 1: Website goals of some organizations

to accomplish their goals. One possible solution to make such databases accessible through internet using present methodologies and tools can be as follows: design a web structure using some methodological approach and instantiate HTML documents with the contents of database. This will be very complex and not recommended if the database is too large. Also if the database is volatile this will not solve the problem as modifications in the database will not get reflected in the application automatically. So there is a need for a methodology that accommodates other models/methods for information retrieval and data access.

Focusing on the above considerations we have proposed a methodology for the design and implementation of hypertext applications that work as a web interface to an existing database. This methodology is also useful to design other classes of applications that can be designed using present methodologies. We also have developed a WIDB(Web Interface for DataBase)Design tool. This tool provides an environment using which the designer can design web interface to an existing database according to the steps of our methodology.

1.5 Thesis organization

- Chapter2 discusses some popular models and methodologies for the systematic design of hypertext/hypermedia applications. Later the issues that are to be considered in the design of a web interface to an existing database are discussed.
- Chapter3 discusses our methodology for the design and implementation of web interfaces to existing databases.
- Chapter4 discusses about WIDB design tool we have developed for the design of web interfaces for existing databases.
- Chapter5 discusses about the implementation details of the WIDB design tool and the Presentation system which presents data demanded by the user from the database.
- Chapter6 concludes the report and suggests further extensions that can be made to the present system.

Chapter 2

Related work

In this chapter we will discuss some popular data models and methodologies proposed for the systematic design of hypertext/hypermedia applications. Later the issues that are to be considered in the design of web interface to an existing database are discussed.

2.1 Hypertext Data Modeling(HDM)

HDM[6] is a data model proposed to describe hypertext applications. According to HDM terminology a hypertext application can be divided into two portions: a hyperbase and a set of access structures. A hyperbase consists of the HDM elements: entities, components, units and different types of links. Access structures serve the reader to properly select an entry point for further navigation. In addition HDM mainly focussed on modeling the hyperbase. The HDM primitives for modeling hyperbase are as follows:

1. **Entities and Entity types:** An entity is a smallest autonomous piece of information which represents some real world object of the application domain. Entities are naturally grouped into Entity types.

2. **Components:** An HDM entity is a collection of components arranged in a tree like fashion. Components are in turn made of units.
3. **Unit:** A unit corresponds to a component associated with an entity. Units are the smallest chunks of information which can be visualized as nodes in hypertext notion.
4. **Links:** HDM supports three types of links. Presentation links interconnect units corresponding to the same component. Structural links connect components belonging to the same entity. Application links represent domain dependent relationships among entities.

HDM shares some apparent similarities with Entity-Relationship(E_R) model[10]. The HDM model can be used as a modeling device in the conceptual design of a hypermedia application. Even though HDM model is effective in its own respect, do not constitute a well defined, consistent development environment, in particular the navigation aspect. HDM does not describe a method for design and development of hypermedia applications.

2.2 Relational Management Methodology(RMM)

RMM[14] is a seven step methodology for the design and construction of hypermedia applications. RMM's data model is an extension if E-R model and is shown in figure-2 [14]. The upper part of figure shows the domain primitives and are useful in modeling the information in the application domain. Entity types and their attributes represent abstract or physical objects. Relationships describe associations among different entity types. An entity may be described as a collection of slices and attributes. Navigation is supported in RMM by the six access primitives shown at the bottom of figure-2. The grouping construct is menu like mechanism that enables access to other parts of a hypermedia application. The seven steps of the methodology are given below:

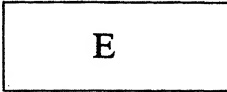
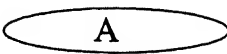

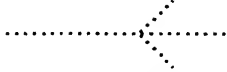



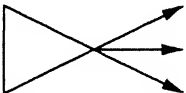
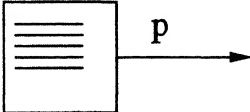
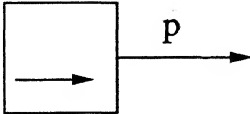
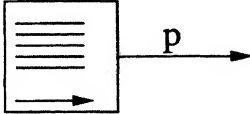
E/R Domain Primitives	Entity	
	Attribute	
	One-One Associative relationship	
	One-Many Associative relationship	
RMD Domain Primitives	Slices	
Access Primitives	Uni-Directional	
	Bi-Directional	
	Grouping	
	Conditional Index	
	Conditional Guided Tour	
	Conditional Indexed Guided Tour	

Figure 2: RMM Modeling Primitives[14]

1. **ER design:** This step is to represent information domain of the application using the primitives entities, attributes, slices and relationships - similar to E-R diagram. For each entity its slices and key attributes must be specified.
2. **Slice design:** In the slice design the designer can describe each slice introduced in the ER design step in detail. Also the designer can add cross links between different slices belonging to the same entity. This allows navigation from one slice node to another slice node.
3. **Navigational Design:** In this step designer has to specify the navigational paths between entities via some access structure based on relationships between them. Next is to describe groupings that act as initial hypernodes from which the other nodes of the application can be traversed. For each group the designer has to specify the entities that are to be included in its menu.
4. **Conversion:** The conversion rules to transform the RMM data model primitives to objects in target platform are not yet completely defined.
5. **User Interface Design** involves the design of screen layouts for every object appeared in ER-design.
6. **Runtime Behavior Design** involves populating the hyperbase with some instances and testing the prototype. This step makes sense only when building hypermedia application using a tool that follows the methodology step by step in the design of application.
7. **Implementation** is construction of final application.

2.3 Systematic Hypermedia Application Design with OOHDM

Object Oriented Hypermedia Design Method(OOHDM) [4] is a four step methodology for the design of hypermedia applications. OOHDM's data model [5] is an

extension of Rumbaugh's Object Modeling Technique(OMT) diagrams[11]. The OOHDM steps are as follows.

1. **Conceptual Design:-** During conceptual design a model of the application domain is built using OMT principles having conceptual classes and associations. Each conceptual class may be built using aggregation, generalization/specialization hierarchies.
2. **Navigational Design** Navigational design is expressed in two schemas: the navigational class schema and navigational context schema. In the navigational class schema the navigational objects of the hypermedia application are determined from the conceptual model of step1. The navigational objects reflect the chosen view over the application domain. The classes of this schema represents nodes of the final application. The navigational classes that OOHDM supports are nodes, links and access structures such as indexes and guided yours. Nodes are object-oriented views of conceptual classes defined during conceptual design. Links represent associations in the conceptual schema.

In the navigational context the navigational space is structured accordingly. Each navigational class(node, link or access structure) of navigational class schema will be defined a navigational context describing its navigational behavior.

3. **Abstract Interface Design** OOHDM uses the Abstract Data View (ADV) design approach for describing the user interface of a hypermedia application [3]. Abstract Data Views are formal, object-oriented models of interface objects and they are specified by showing:
 - The way in which they are structured using aggregation and generalization/specialization as abstraction mechanisms.
 - ADVs allow defining the interface appearance of navigational objects and of other useful interface objects (such as menu bars, buttons and menus).
 - The way in which they are statically related to navigation objects. We use Configuration Diagrams as a diagrammatic tool for expressing these

relationships.

- How they behave when reacting to external events; in particular how they trigger navigation, and which interface transformations occur when the user interacts with the application.

4. Implementation

To obtain a running implementation, the designer has to map the navigational and abstract interface models into concrete objects available in the chosen implementation environment.

2.4 Browsing A Database

The major methods of accessing data in current database systems are querying and browsing. Browsing within a database as best exemplified by hypertext systems, consists of viewing a database item and linking to related items on the basis of some attributes or attribute value. The issues that are to be considered while developing hypertext applications that behave as an interface to browse the database include the following [9][12].

1. The user may not be familiar with the principles employed by the system to organize the data(the data model).
2. The user may not be familiar with the contents or definition of the particular database to be accessed.
3. The user will not be proficient in the procedures used for definition and retrieval of required information(the data language).
4. The user may have only a vague retrieval target(e.g., the user is looking for something "interesting" or "suitable").
5. The user may have a clear retrieval target but lacks some of the information necessary to describe it(e.g., the user wants to find out the meaning of a word

in a dictionary, but can not spell it correctly). To incorporate such types of browsing AI techniques may be necessary.

6. The designer may like to provide multiple views of the same domain of information(e.g., photographs indexed by name, date, photographer, location etc).
7. There should be efficient access mechanisms to provide effective and fast access even if the domain of information is large.

Covering the above issues we propose a methodology for the systematic design of hypertext applications that act as a browsing interface to an existing database. We also developed a WIDB(Web Interface for DataBase) Design Tool that helps in the design and implementation of such applications to view a database.

Chapter 3

Methodology for Design of Data intensive Web Sites

In this chapter we present in detail our methodology for the design and implementation of hypertext applications that act as web interfaces to existing databases. We illustrate the methodology with an example. To begin with, here are a few concepts.

3.1 Data Model

A data model is a collection of conceptual tools to describe an abstract view of any real world systems. The modeling primitives used in the methodology for describing the components of the hypertext application are shown in Figure-3. This data model was derived from the well known data model, ER-model, which is used to describe the conceptual view of a database.

3.1.1 Structural Primitives

The structural primitives of this data model in the first half of figure-3 are useful to specify the conceptual schema of the application .

- The Entity ¹ can be used to represent a conceptual node (typed node) in the

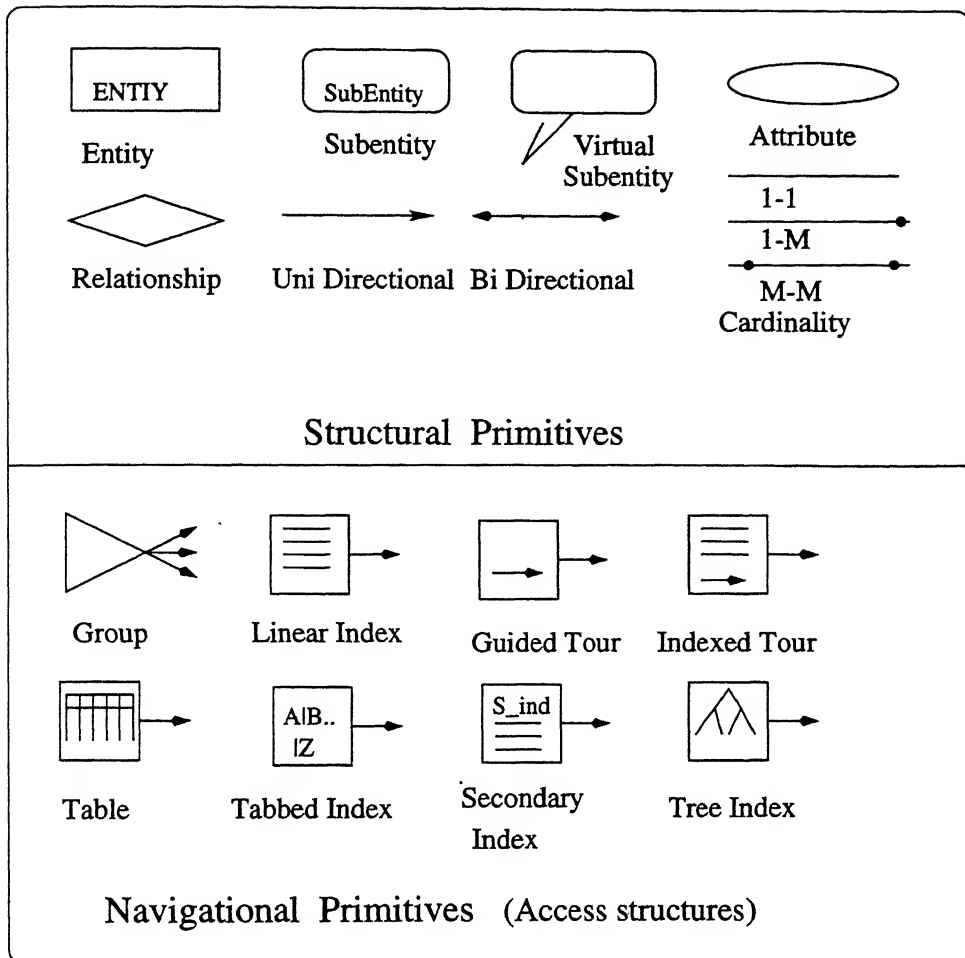


Figure 3: Modeling Primitives

hyper space of the application to be designed. The represented hyper node must be autonomous describing a single concept or idea. Here the Entity node can be a composite node so that a group of nodes can be represented as its subnodes in a hierarchical structure.

- A Sub_entity represents a subnode under the hierarchy of an Entity node and thus useful to organize internal structure of an Entity node as an hierarchy of subnodes may be part of parent node or fully dependent on the parent node. A Sub_entity can be one of the two types:

¹It is to be noted that an Entity may represent a type node or an instance node depending on the context.

1. a normal Sub_entity that will be linked to its parent node through an organizational link in the final application.
 2. a Virtual Sub_entity where all of its structure will be embedded into its parent hyper node and thereby it will not be shown as a separate hyper node in the final application.
- An Attribute represents the atomic piece of information and is part of an Entity node or Sub_entity node.
 - The Relationship represents the conceptual tie between the Entity nodes i.e the reachability from one Entity node to another. The Relationships can be unidirectional or bidirectional. Depending on the directionality the navigation to and fro between the source and destination Entities will be provided. The cardinality of the Relationship can be one to one, one to many or many to many.

The labels of the Relationship can be used to associate with the links in source and destination entity nodes.

3.1.2 Access Structures

The Access Structures in the second half of Figure-3 are useful to specify efficient and fast access mechanisms for the instances of Entity nodes directly and for the nodes that are linked to these nodes through Relationships.

The group construct is used to specify an initial node from which the other parts of an application can be accessed. This acts as a menu containing all the Entities as its items. By selecting one, one can access the objects of that Entity. More about these primitives will be discussed in the methodology.

3.2 Methodology

The methodology is organized into a sequence of seven steps that are as follows: Conceptual modeling, Physical mapping, Navigational modeling, Conversion into SNS templates and HTML templates, User Interface modeling, Functional modeling and Implementation. Starting with the first step the designer can cycle through all steps at his/her ease and convenience.

3.2.1 Conceptual modeling

This step captures the conceptual views of the web interface to be designed by accepting the structural diagram from the designer. This structural diagram can be specified graphically by making use of our modeling primitives shown in Figure-3. In this step the designer can follow the following steps for a systematic specification of the structural diagram.

1. Modularize the data space:- This step encourages the designer to modularize the whole data space into syntactic units in such away that
 - a unit represents a single concept that needs to be referenced elsewhere
 - it is independent of existence of other nodes.

Each syntactic unit can be symbolized as an Entity node of our data model. This process of identifying syntactic units is not unique and is a difficult process. But in the case where the target application is to show a database this will be very simple as that would have been modularized as Entity nodes already.

2. Determine the Relationships between the Entity nodes that are determined in the previous step. These Relationships represent the referential links across the instances of source and destination Entity nodes in the final application. These links represent the interdependencies among the hyper nodes of the

applications. The Relationships may have some atomic properties represented as Attributes.

By going through the above steps repeatedly and suitably refining the Entity nodes and their Relationships the designer comes to a stable structure.

3. Select each Entity node and find the internal structure of its information. An atomic piece of information related to this Entity node can be specified as an Attribute. An Entity node can again be described as a hierarchy of subnodes, which in turn are represented as Sub_entities in our data model. A Sub_entity is a collection of closely related information that is related to the parent Entity in one of the two possible ways.
 - one is that the subnode is part of the parent node (aggregation). For example Car as parent Entity node and description of its Parts as Sub_entities.
 - The other is that the subnode is fully dependent on the parent node (weak Entity). Example :Employee-Children is one such example where an instance of Children can exist autonomously only with its parent Employee key value.

These subnodes will be mapped into hyper nodes and connected by organizational links in the final hypertext application. Generally the designer may desire to specify the internal structure of the Entity as a hierarchy of Sub_entities (to reduce complexity of the structural diagram of the root Entity or for the reason discussed in next section) and wants the Sub_entities information to be shown in the hyper node of parent Entity.

For this we added another Sub_entity type named virtual Sub_entity to our data model. This specifies that the corresponding Sub_entity's information will be appended to the parent Entity information in the final application. Thus each Entity and Sub_entity can be zoomed into and can be described in a systematic way.

The above steps can be repeated to refine the conceptual view/model of the hyperbase.

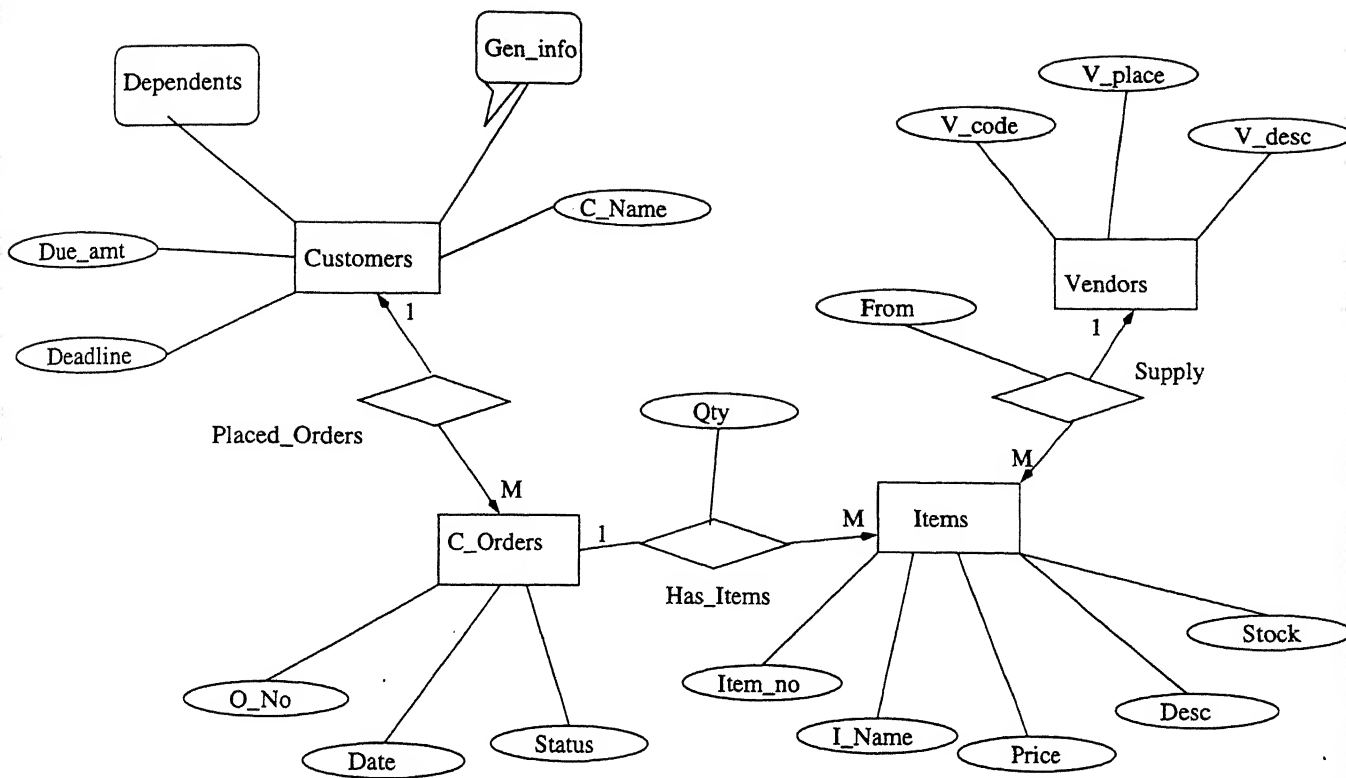


Figure 4: A Conceptual model for A Departmental Stores

As an example of the application of the above methodology consider the case of a Departmental Store (figure-4). In this example a Conceptual model of the departmental store has the following aims to model. The store wants to bring its data and services available to customers through web access, so that customers can shop the items and place their orders online. In the model vendors, items, customers, orders are shown as autonomous Entities. The Relationships supply, placed-orders, has-items represent associations between Entity instances. The customers Entity is shown as a hierarchical tree of two Sub_entities. In these general-info is virtual Sub_entity i.e., they must be appended to customers entity node. while Dependents Sub_entity nodes will be linked from customers entity nodes.

3.2.2 Physical Mapping

The Methodology should capture all the required information systematically and make that available for implementation. In the case where the target application is an web interface to an existing database, the implementation part will involve writing a set of scripts which accepts the user requests and and presents the data from the database. These scripts are need to be modified whenever the database properties changes.

So it is better if the methodology capture the physical mapping of the conceptual model with the physical database and makes them available in the Structural and Navigational Specification(SNS) templates generated in the fourth step. Once we have this information in SNS templates then the implementation part will nothing if we have the scripts that present data from database with the aid of information available in these templates. These scripts will work properly with any of such application with out the need of modification.

Moreover this step will encourage the designer to describe the conceptual model with whatever domain names he feel appropriate and without bothering about the physical properties of the data base. Thereby it is also possible that an Entity node may span more than one table in the database or a table may contain information about more than one conceptual Entity node. A mapping form for Entity/Sub_entity, Relationship are shown in figure-5.

3.2.3 Navigational Modeling

The distinguishing key feature of hypertext/hypermedia applications is the notion of navigation possible by inclusion of hyper links in the document. But the dilemma of large hypertext systems is to navigate selectively and quickly in the abundant information domain. The provided Navigational possibilities should enable the user to explore the information(may be in multiple ways) without getting disoriented in the complex hypertext system.

The Access structures of the Data model shown in figure-3 can be used to provide

Physical Mapping of Items

Name:
Items

Alias Name

Items

Access Information

Attributes

Domain Name	Alias Name
Item Number	<div style="border: 1px solid black; padding: 2px 10px;">Item_no</div>
Item Name	<div style="border: 1px solid black; padding: 2px 10px;">I_name</div>
Item Price	<div style="border: 1px solid black; padding: 2px 10px;">Cost</div>
Present Stock	<div style="border: 1px solid black; padding: 2px 10px;">Stock</div>

Figure 5: A Physical Mapping of Items Entity

access for the instances of an Entity node from a Group or another Entity node to which these are related through some Relationship.

Our navigational modeling starts with a default model derived from the Conceptual model. The initial model contains all Conceptual Entity nodes. the Relationships among the Entity nodes will be represented as a set of Meta links via some Access structures. Here a Meta link is a link type between typed nodes. These Access structures can be simply linear indexes and provide access from a source entity node to its related nodes of the destination Entity. The number of links derived from a Relationship depends on its directionality and the Access structures depends on its cardinality.

- Example:- The Supply:one-many & bidirectional, Relationship of the above Conceptual model shown in the figure-4 is represented in the Navigational model (figure-6) as two links, one from Vendors to Items with an intermediate access structure: linear index and a direct link from Items to Vendors with labels Supply and SuppliedBy correspondingly.

Finally a Group node (a starting node from which the user can navigate to other parts of the application) will be added to this initial model.

Starting with this initial model the designer has to determine the Entity nodes that need to be placed in the Group node. For each Entity that is included into the Group node the designer can provide effective access to its instance nodes by selecting appropriate Access structures, these are shown in figure-3. The designer can provide multiple views of the same domain of information by selecting multiple Access structures to the same Entity node.

- For example in figure-6 the navigational model contains two Access structures to the Items Entity. Thus an user can navigate the Items instance nodes by selecting one of the Tree Index or Secondary Index on Item Name.

Similarly the access structures that represent Relationships between Entity nodes can be replaced with any other Access structure to provide effective navigation. Moreover by defining multiple groups, different navigational models can be built for the same application. One more point is that these access structures can be pre-conditioned to narrow down the domain of data to which they apply their access mechanism. The different access structures are as follows.

- Group: This serves as an initial node to access other parts of the hypertext document.
- Linear Index: This can be the default access mechanism provided to the Conceptual nodes from the main group. This allows navigation from the index to any element and from element to the Index.

- **Guided Tours :-** Guided Tour allows to navigate all the elements sequentially to and fro.
- **Indexed Guided Tours:** The user can have both Indexed and sequential access facilities.
- **Table format:** If the application being developed is a web interface to an existing database then a simple Entity with small number of Attributes can be preferred to be shown in a table format.
- **Tabbed Index:** If the index to be shown is too large then Tabbed Index can be used to narrow down the domain of information the user is interested in. However when the distribution of Key Attribute's domain is uneven then Tree Index is preferable.
- **Secondary Index:** This facilitates faceted retrieval of data. Example:- Students indexed by Roll number, Name, email address etc. Photographs indexed by Date, Photographer, location, topic etc.
- **Tree Index:** If the information domain is too large then Tree Index is the best access structure to access an element quickly in a simple manner.

A navigational model for the Departmental store example considered earlier in figure-4 is shown in figure-6.

3.2.4 Conversion into SNS templates

Each conceptual Entity and Sub_entity node will be mapped into Structural and Navigational Specifications(SNS) Templates. A SNS template is a structured node containing meta data about the structure of the node (i.e its Attributes, Sub_entities), its physical mapping information, Access mechanisms determined in the Navigational model to provide access into its items and Links derived from these Access structures.

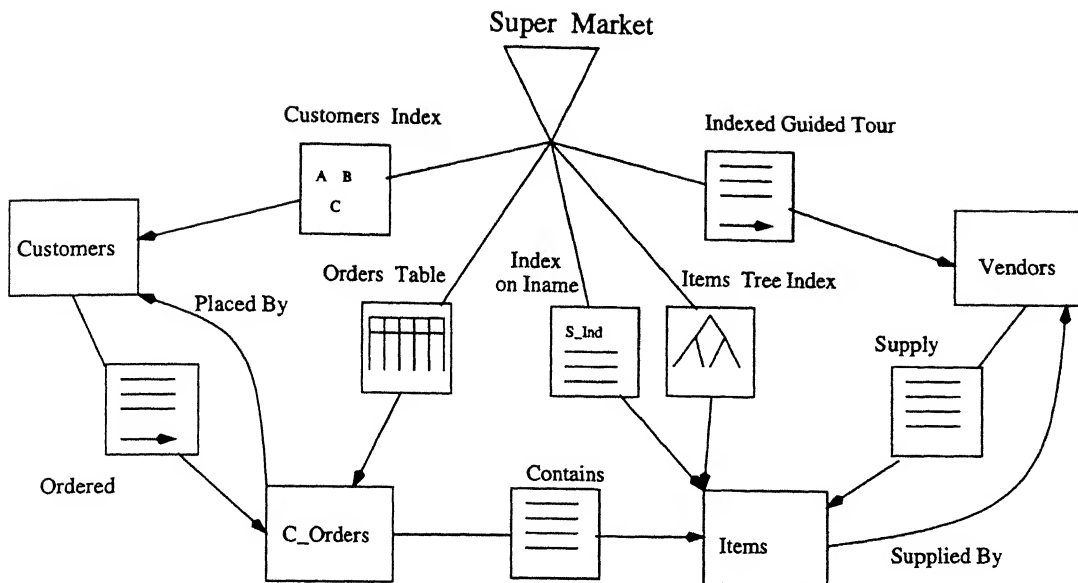


Figure 6: Navigational Model for Departmental Stores

Each Relationship of the conceptual model will add Link attributes in the source and destination Entity's templates depending on the directionality of the Relationship. A Link attribute consists of label of the link and its target SNS template. Then a SNS template will be generated containing meta data about its Attributes, physical mapping and Access mechanism specified for this Relationship in the Navigational modeling. These templates will help in the automation of implementation part of the methodology for a class of hypertext applications.

3.2.5 Interface Modeling

The Interface Modeling encourages the designer to focus on screen layout of HTML documents, searching tools, Metaphors for visualization, querying, zooming etc. to make documents more perceptible to the user. But the boundaries for user interface design are always not clear.

Again our Interface modeling starts with a default Interface model. Associate an HTML template with each of SNS Template and Group Nodes determined in Navigational model. Each HTML template consists of Attributes as fields, all kinds

of links: links to other HTML nodes, "prev", "next" etc with some default layout (linearly arranged an example).

Now the designer can arrange his preferred layout for each HTML template using one of the many tools available commercially. As information hierarchies are the most frequently used metaphors, the Tree Index access structure can justify this type of visualization. The search tools can be added in any of the Templates.

As the complexity of the application increases it becomes distressingly easy for a user to become lost or disoriented. To solve this problem Overview diagrams may be included in a separate frame, as these serve as excellent navigation aids. Global view diagrams provide overall picture of the hypertext application and serve as anchors for local view diagrams. Local view diagrams provide a fine-grained picture of the local neighborhood of a node. After the User Interface design the designer is ready with SNS Templates, HTML Templates and input forms for search tools.

3.2.6 Functional Modeling

As Internet became cheaper and faster media for communication there is tight competition from many organizations like public service providers, Products, departmental stores etc.. to make their services available through WWW. As email is not a good solution to provide services on large scale routinely, there is a need to incorporate those services as functionalities in their web sites. These services may have to do some computing, accepting customer orders, updation of data by authorized users, etc.

- In a Departmental Stores processing a customer order include: Accepting an Order, Checking validity and availability, Entry into the orders file and mailing the receipt to the customer.
- A vendor may like to place his products in the Departmental Stores.
- A person may like to see the gross value of his stocks under a company.

So in this step for each functionality to added into the application the designer has to specify the following.

- the signature of the functionality i.e., name and its arguments.
- the input parameters to be read from the user. From this an input form can be generated. to read these parameters when the user activates the functionality.
- the description label that needs to be associated with the link that invokes this functionality.
- the target HTML node in which this functionality should be made available.

In the implementation part for each functionality a script can be generated, which does the following.

- reads the input parameters from the user if any.
- writes the HTML header.
- calls the the functionality.
- writes the HTML trailer.

3.2.7 Implementation

Implementation of the hypertext application can be done by directly instantiating the HTML nodes according to the specifications in SNS templates and HTML templates. For this process a tool, like Intermedia can be used to instantiate the template with given data. If the target application is to duplicate a database then a tool can be developed to instantiate the web documents with the data from database with the aid of HTML templates and SNS templates. But if the application is an interface to an existing database then few simple scripts are to be written which access both the HTML and SNS templates and present the data demanded by user according to the specifications in the templates.

Chapter 4

Web Interface for DataBase(WIDB) Design Tool

We have developed a tool for The automation of the design and implementation of Web Interface to view an existing database. This tool provides an environment using which the designer can design such web interfaces according to the steps of our methodology. The tool is platform independent as it was developed in Java and the scripts provided are written in Perl. The simplicity of this tool is that given only structural diagram of the database this can generate the navigational possibilities and an interesting interface with search options. Two steps of the methodology (User Interface modeling and Functional Design) are not implemented in the tool. The designer can cycle through the implemented methodological steps in any order.

The tool allows to do the following:

- The designer can draw the conceptual view of the web interface structure to be designed as a diagram.
- The designer can zoom into each Entity and Sub_entity and can describe its internal structure graphically.
- The required navigational possibilities can be specified graphically.

- The tool will convert the design specifications into a set of HTML documents and SNS templates.
- There is a Save option to save the design specifications in a file, so that it can be loaded later for further modifications.

The functioning of the tool is explained below in detail.

4.1 Conceptual Modeling

This step facilitates the designer to describe the conceptual view of the web interface structure graphically using the following design objects and thus captures the characteristics of the application.

- An Entity shown by a rectangle represents an autonomous Conceptual node (Type node) describing single concept or idea.
- A Sub_entity shown by a rounded rectangle represents a subnode in the hierarchical organization of an Entity node. A Sub_entity can be Internal (i.e Virtual) or External (i.e it will be linked from the parent Entity node).
- Relationship shown as a diamond is used to connect the related Entity nodes of the application.
- An Attribute shown in an oval is used to represent an atomic unit of information that belongs to either an Entity/Sub_entity or Relationship.

The structural diagram that has been described with the use of these Design constructs by the designer need not be the same as the ER-Diagram of the database. Moreover, the domain names of the Entity, Sub_entity nodes, Attributes, Relationships need not be the same as in the physical database. These can be framed in more readable and understandable form.

There is also an option to zoom into each Entity or Sub_entity to describe its internal structure separately. Thereby the complexity in the main structural diagram

can be reduced. The figure-7 shows a structural diagram described using this tool. The figure-8 shows the internal design of Dependents described by zooming into it.

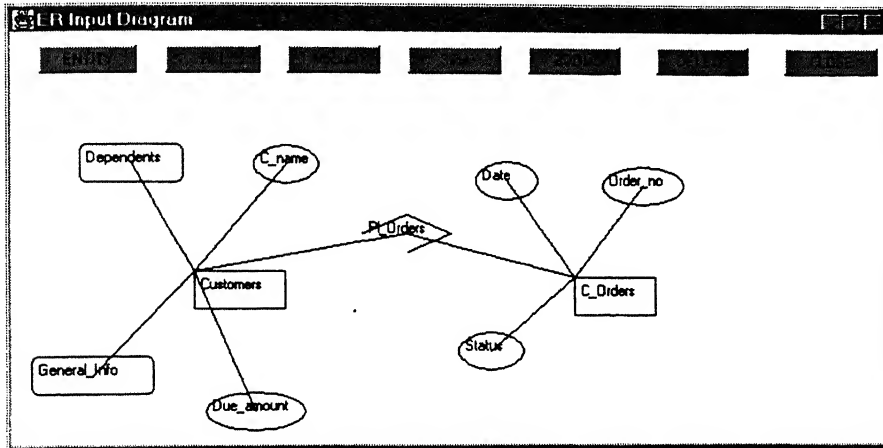


Figure 7: Conceptual Modeling using WIDB Design Tool

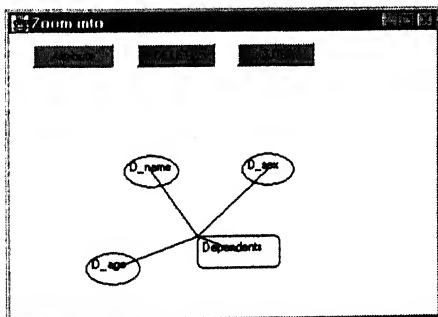


Figure 8: Zoom into Dependents Sub_entity

The physical mapping of the Conceptual model with the database is merged in this part itself. Whenever the designer adds a design object it will prompt for its physical mapping with the database. For each Entity, Sub_entity, and Relationship added it asks for alias name(i.e table name) and Access information(i.e database, user and Password etc). Similarly for an Attribute it prompts for its alias name in its physical table. If an Entity information spans more than one table then for each set of Attributes belonging to a table, the designer can add a virtual Sub_entity having these Attributes.

4.2 Navigational Modeling

This part starts with an initial Navigational model that was derived from the Conceptual model. In this model there will be one Main Group which serves as a starting node to access other parts of the application, Conceptual Entity nodes that are connected from Main Group through the Access structures with linear indexing mechanism. The Relationships between the Entity nodes will be represented as one(if unidirectional) or Two(if bidirectional) Meta links between source and destination nodes. These links are implemented via an access structure if the outgoing cardinality is one-many(with linear indexing mechanism).

For example in figure-9 the Placed-Orders Relationship is represented by two links, one from Customers to Orders via linear index Access structure with Ordered label, and the second is a direct link from Orders to Customers with label OrderedBy.

Starting with this the designer can add/delete the access structures that provide access into items of Entity Nodes from the Main Group. The designer can also add Secondary indices from Main_Group into the Entity nodes. The supported Access Structures are Linear Index, Guided Tour, Indexed Guided Tour, Table format, Tabbed Index and Secondary Index. In figure-9 the Orders is reachable through Indexed Tour and Customers is reachable through Tabbed Index.

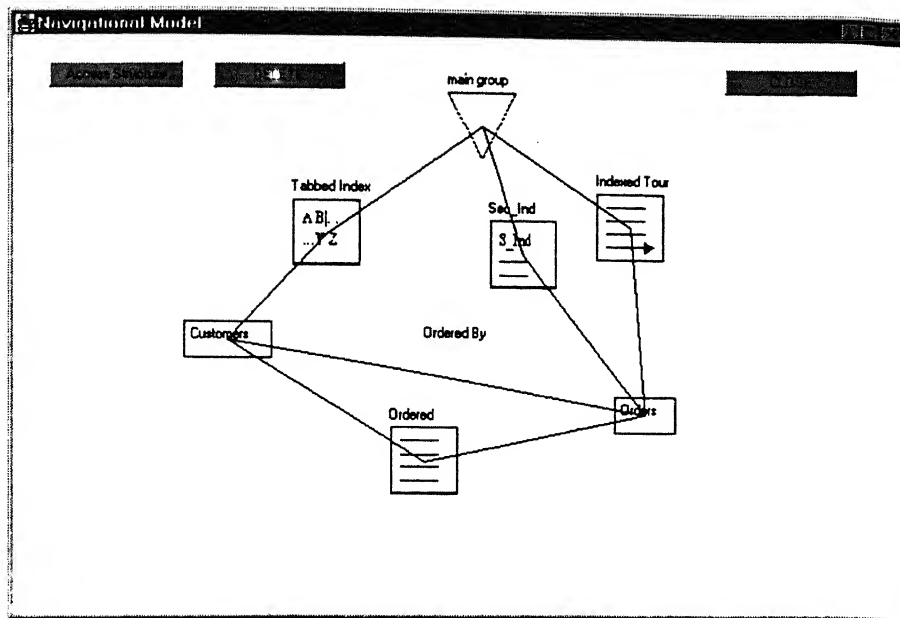


Figure 9: Navigational Modeling using WIDB Design Tool

4.3 Conversion into SNS Templates

In this part a set of HTML documents and SNS Templates will be generated. The list of HTML documents are as follows:

- entities_toc.html: This contains all the links to entities that are in the Main Group.
- R_ships.html : This contains links to all the Relationships documents.
- For each Relationship a document to show source and destination entities in side by side frames will be generated.
- For each Entity a form to search in that Entity's instances will be generated.

For each Entity, Sub_entity, and Relationship a SNS Template will be generated. The structure of the template for the Customers Entity is as follows:

Name: Customers
Type: Entity
Alias: Cust
Access: database=lib&user=scott&pass=tiger
From: null # From Entity In case of Relationship
To: null # To Entity In case of Relationship
Itype: Linear Index # This is Indexing Machanism
Next: false
Prev: false
Main: true
First: false
Contents: false
No_atr: 4 # No of Attributes
C_name:string:Customer Name
Addr:string:Address
Phone:Integer:Phone Number
Due_amt:real:The Due Amount
[C_name] # The Primary Key Attributes
L_attrs: 2 # No of Link Attributes
Ordered:Pl_Orders.tem:false:from

4.4 User Interface

The Interface of the application generated from this will be as shown in the Results-PS-1. The multiple frames will avoid disorientation while navigating. Another interesting feature of this interface is that the Entities that are related through a Relationship will be shown in side by side frames to provide a better look and easy navigation in both directions. A sample look of navigation through Placed_Orders relation is shown in Results-PS-2. Another facility is the search options in each of

the Entity nodes. The search tool for Customers is shown in Results-PS-3.

4.5 The Save Option

There is an option to save the design specifications of the application in Conceptual modeling and Navigational Modeling. This will serve as documentation and can be modified later according to the new requirements.

Chapter 5

IMPLEMENTATION

The previous chapter described about the WIDB Design Tool we developed for the design and implementation of Web Interface to view an existing DataBase. This chapter discusses about the implementation details of the various components of the WIDB design tool and about the Presentation System that presents the data demanded by the user from the database. The WIDB design tool has been implemented using Java[8] and the Presentation System has been implemented in Perl[15].

5.1 Implementation of WIDB Tool

The runtime architectures of the WIDB design tool and the Presentation System are shown in the figure-8. By using the WIDB tool the designer describes the structural specifications and Navigational specifications through the interface of the tool in the Conceptual and Navigational modeling steps. The Input Processing unit(as shown in figure-8) captures these specifications from the tool's interface and generates a set of HTML documents(which act as a web interface to navigate the database) and a set SNS Templates(which the Scripts will access to present the data demanded by the user through the web interface generated).

The component-action hierarchical diagram of the interface part of the WIDB

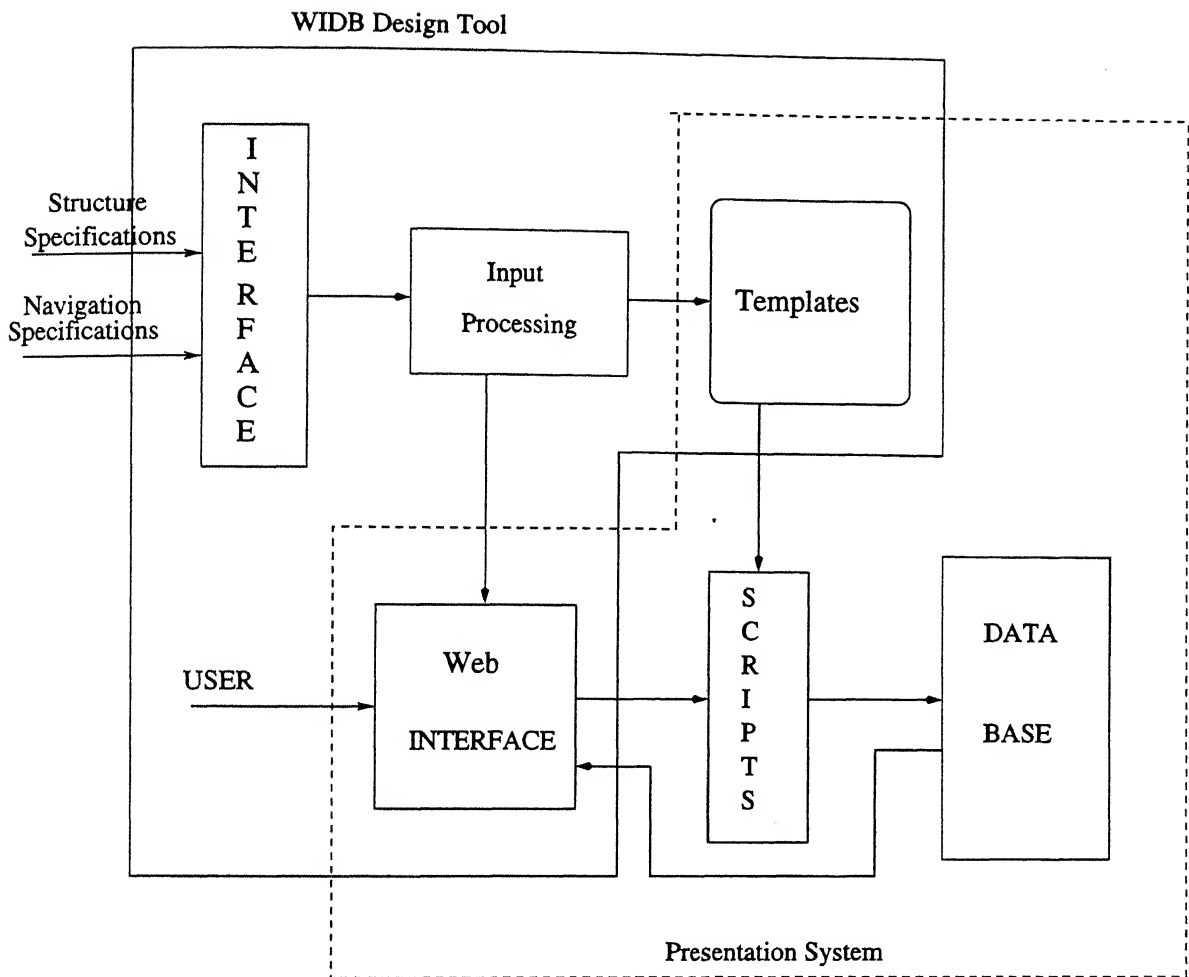


Figure 10: Runtime Architecture of WIDB Tool and Presentation System

design tool is shown in figure-9 ¹.

The main menu bar (Appmenu) contains three pull down menus. The FILE menu contains three MenuItems : New(To start a new application from scratch), OPEN(To Load an Existing file), EXIT(Exit with SAVE option). The CONTEXT menu contains three MenuItems:

- Conceptual Model will bring the ER_Window to the user to describe the Conceptual model with the use of Buttons and help provided in the HELP menu.
- Navigational Model will bring Nav_Window to model navigational possibilities.
- The Conversion menuItem will bring Main_page_window to read the URL, title of the Main Page.

The HELP menuItem will bring up Help_Window with guidelines to use the tool. All these windows are shown in the Results placed after Chapter-5.

The designer creates a web interface using this tool in two ways. He can use the OPEN menuItem to load an external file and can modify it. He can also start from scratch by selecting NEW menuItem. After finishing Conceptual Modeling and Navigational Modeling the designer can now do Conversion to generate a set of HTML documents and SNS Templates. Finally while exiting the tool the designer can save the design specifications of the application. The classes other than the interface components are as follows:

- Basic_Entity: This class captures all specifications about an Entity/Sub_entity node.
- R_Ship: This class captures all specifications about a Relationship.
- Template: This is a SNS Template containing internal structure and navigational possibilities of an Entity, Sub_entity, or Relationship.
- attr This represents an Attribute.

¹An Arrow indicates that its right hand side component will be shown on the selection of the left hand side component. The tree structure describes the class hierarchy of that component.

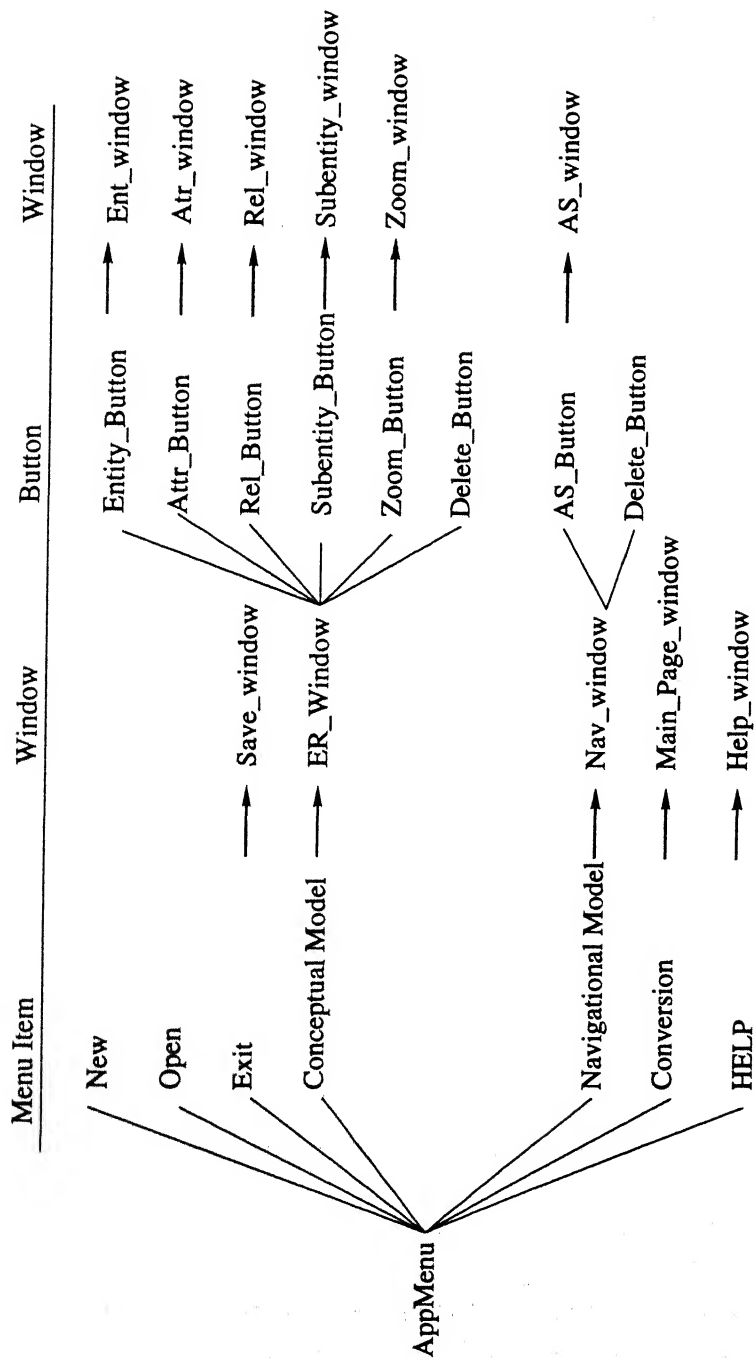


Figure 11: Runtime Architecture of WIDB Tool and Presentation System

- `input_p`: This provides a set of functions to process the input options selected.
 - `convert()`: It does the conversion into SNS templates.
 - `load_File()`: It loads the external file containing design specifications of an application.
 - `generate()`: This generates a set of HTML documents and writes the SNS Templates to files.

5.2 Implementation of Presentation System

The presentation system is used to navigate through the hyper space containing Entities and Relationships. The user first accesses the main page containing three frames as shown in Results-PS-1. The top left frame contains an index of links. By selecting one of these a search form of that Entity will be shown in the top-right frame. The user can navigate the instances of that Entity by selecting the bottom link in that form. The bottom frame contains another index of links. By selecting one of those the user can view the Entities in that Relationship in side by side frames as shown in figure Results-PS-2. So by using the interface of the presentation system the user can navigate through all the Entities and Relationships. The runtime architecture of the presentation system is shown in the figure c-a diagram.

The navigation through the Entities and Relationships in the presentation system is achieved by the Scripts written in Perl. These Scripts retrieve the data requested by the user from the database and presents it according to the specifications in SNS templates. Here these scripts can be classified into two types.

- TYPE-I Scripts:-

The first type scripts provide access to the instances of Entities and from one entity node to all its other related entity nodes. This is similar to moving from a vertex to one of its adjacent vertices in a graph. A brief description about the functioning of these scripts is as follows:

- Script1.cgi:- This script will be invoked when the user wants to navigate the instances of an Entity. The input parameter to this script is Template name of the Entity. The script finds from the Entity's template the access information of the Entity, its structure, the indexing mechanism and retrieves the Entity's data from the database and presents according to the indexing mechanism.
 - * Example: If the index_type is Guided Tour then it will show the first record with links to other related entity nodes.
 - * If the index_type is linear Index then it will show all primary key values of that Entity as links.
 - * If there are more than one index_types then all the index_types will be shown to user to select one.
- Script2.cgi: This script accepts Template name and Key values of an Entity and presents the records with these key values. Along with the Entity's record it will show all links to the records of other Entity that are related to this through some Relationship. It will also show a link to its search form. This script takes over the job of presentation only after the user initiates a search or navigates the instances of an Entity.
- Search.cgi: When the user clicks on an Entity in the top-left frame then a form will be shown to initiate a search in that Entity's records. When the user presses the submit button after filling the form this script will get invoked with Template name of that Entity and Search values as its input parameters. After successful search it will show an index of links to the records found in the search. Whenever an entity record is shown a link will be provided to get its search form.
- TYPE-II Scripts: When the user selects a link in the R_ships(bottom) frame then two side-by-side frames will be shown in the main frame(top-right). The left frame is meant to show the records of the source Entity of that Relationship and the right frame is meant to show the records of the destination Entity. While doing navigation through the Entities of the selected Relationship the

scripts have to remember the Relationship they are showing and must show only those links of an entity record that point to the Entities of this Relationship. Moreover depending on the type of record to be shown the Scripts have to redirect the output to the appropriate frame.

For example in navigating Supply Relationship the Vendors records will be shown in the left frame and the Items records will be shown in right frame. When user selects SuppliedBy link in a record of Items then the corresponding Vendors record must be shown in the left frame. This is shown in the figure Results-PS-3.

To satisfy the above requirements to navigate only the Entities of a selected Relationship another set of scripts are added to the Presentation System. These scripts that are required to navigate only Entities of a Relationship selectively are classified as TYPE_II scripts.

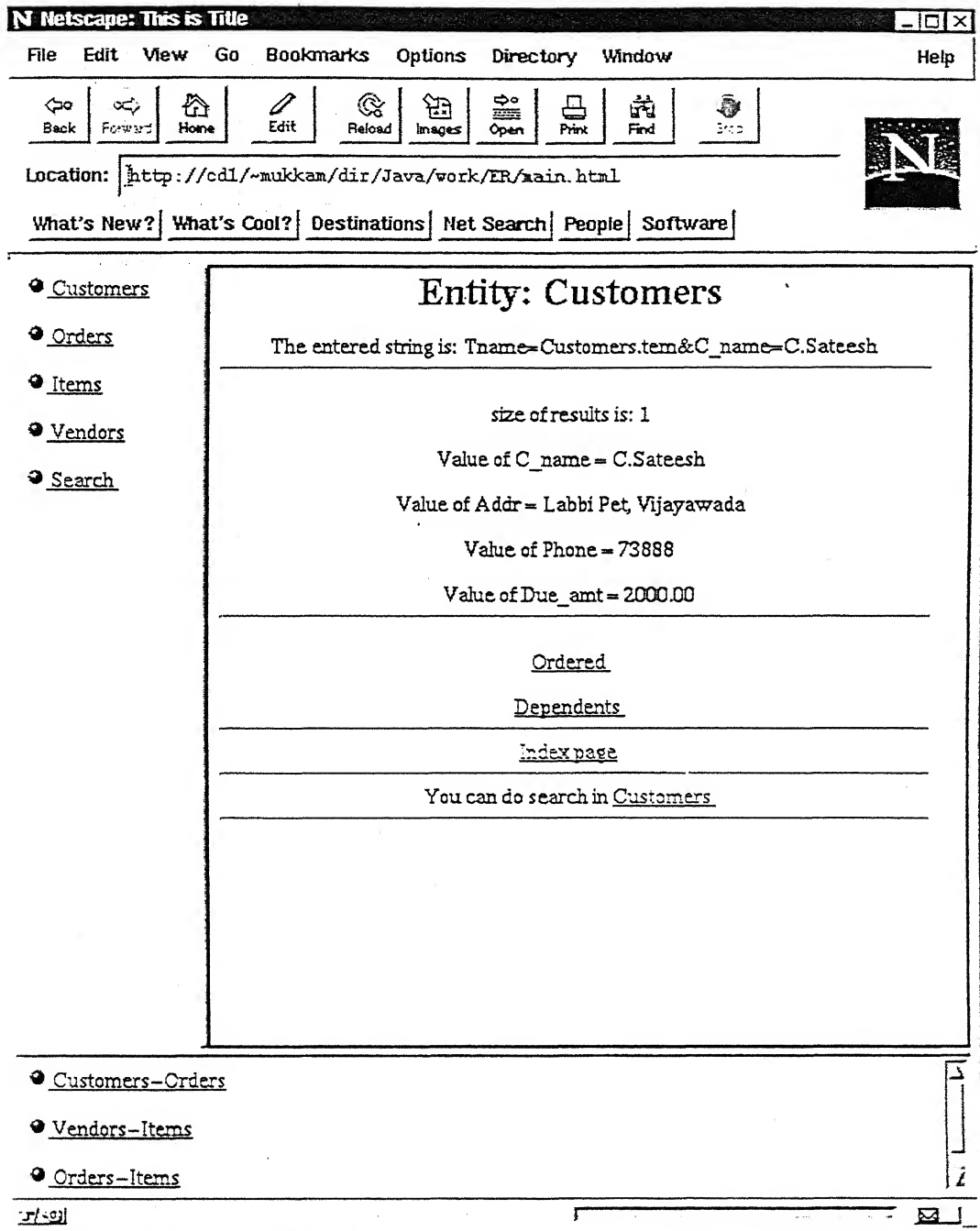


Figure 1: Results-PS-1: Web Interface of Departmental Stores generated by the Tool

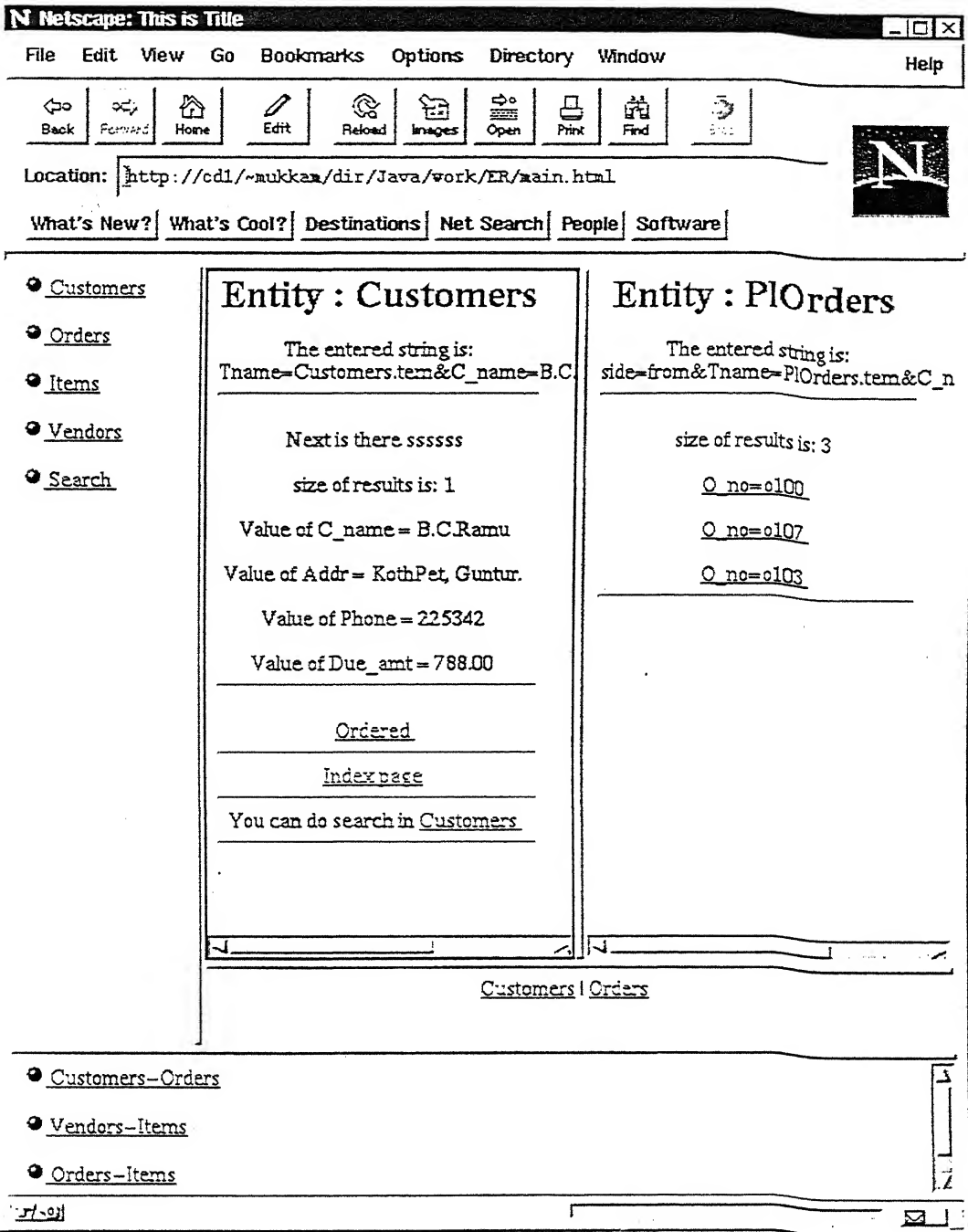


Figure 2: Results-PS-2: Navigation throgh Placed.Orders Relationship

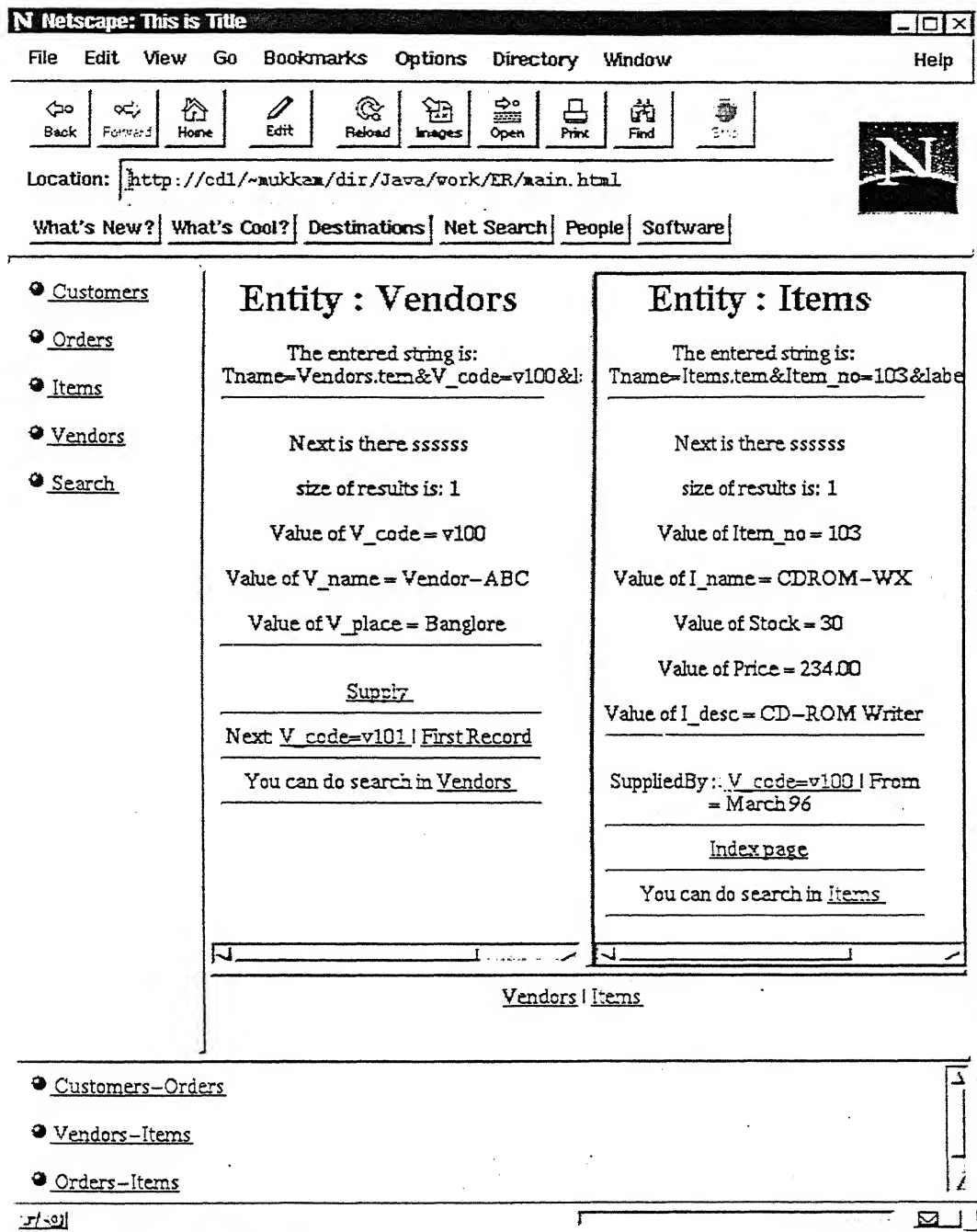


Figure 3: Results-PS-3: Navigation through Supply Relationship

Chapter 6

Conclusions

6.1 Summary

During the past few years the interest in building hypertext/hypermedia applications has been steadily increasing. Unlike traditional information management systems, in a hypermedia system the information is organized as a network of nodes with each node containing a chunk of information. The associations among the nodes of a hypermedia system are established by creating links between them.

As the complexity of the hypermedia systems increase there arises a need for methodologies for the design and development of hypermedia applications. In the recent past some methodologies have been proposed for the design and construction of hypermedia applications that are merely a collection of static nodes interconnected through static links.

In this thesis report we proposed a methodology for the design and development of hypertext applications that behave as web interfaces to existing databases. The data model used in this methodology bases itself on the ER-model. The methodology has been shown to have seven steps:

1. **Conceptual Modeling:** This step helps the designer to determine the required structure of the hyperbase using the primitives entities, subentities,

virtual_subentities, attributes and relationships.

2. **Physical mapping:** This step helps the designer in mapping the conceptual model determined in the first step to the physical database. This encourages the designer to design the conceptual model without bothering about the physical properties of the database.
3. **Navigational modeling:** This step helps to determine the access mechanisms for two types of navigation: to the instances of the conceptual entities and from one entity node to other related entity nodes. This step starts with a default navigational model derived from conceptual model, which the designer can refine according to his requirements.
4. **Conversion into SNS templates and HTML templates:** This step converts the design specifications captured in the above steps into Structural and Navigational Specifications(SNS) templates. And for each conceptual Entity/Sub_entity that appears in the final application an HTML template with a default layout will be generated.
5. **User Interface modeling:** This step helps the designer to make the application more perceptible to the users, by designing screen layout, adding search tools, incorporating visualization metaphors etc.
6. **Functional modeling:** This step determines the services that need some on-the-spot processing on selection, and incorporating them in the application as functionalities.
7. **Implementation** In this step the designer has to write some scripts which present the data demanded by the user according to the specifications in the SNS templates.

We also have developed a Web Interface for DataBase(WIDB) Design tool and Presentation system. The WIDB tool provides an environment using which the designer can construct web interfaces to view existing databases. The tool allows

the designer to design the web interface according to the steps of our methodology. The conceptual model and required navigational possibilities can be specified graphically using the tool's interface. This tool is implemented in Java language. The Presentation system is a set of scripts written in Perl language, using which the user can navigate through the hyperspace. It accepts the requests from users and presents the data demanded from the database according to the specifications in SNS templates.

6.2 Further Extensions

The following extensions can be made to this work:

- The methodology can be extended to include OMT diagrams[11] into its data model.
- User Interface modeling can be studied in detail to facilitate all the possibilities mentioned in our methodology.
- Functional modeling needs a detailed study in order to incorporate functionalities appropriately.
- The tool can be extended to handle multiple level hierarchies in the design of internal structures of entities. Presently, it supports two levels of hierarchy only.
- If the interface modules that interact with the database are implemented in Perl-DBI(DataBase Interface) then the presentation system can work with all types of databases without any need for modification.
- If the data of an Entity is static then its data can be hard-coded into HTML documents. This can be an extension to the WIDB Design tool.

Bibliography

- [1] B.BALASUBRAMANIAN. State of the art review on hypermedia issues and applications. Available At: [http://www.isg.sfu.ca/ duchier/misc/hypertext_review/HT.tar.gz](http://www.isg.sfu.ca/duchier/misc/hypertext_review/HT.tar.gz).
- [2] CONKLIN.J. Hypertext: An introduction and survey. *IEEE Computer* 26, 9 (Sep 1987).
- [3] D.D.COWAN, AND C.J.P.LUCENA. Abstract dat views, an interface specifications concept to enhance design for reuse. *IEEE tran. on Software engg.* 21, 3 (Mar 1995).
- [4] D.SCHWABE, AND G.ROSSI. Systematic hypermedia application design with oohdm. Available at [http://www.inf.puc-rio.br/ schwabe/OOHDM/](http://www.inf.puc-rio.br/schwabe/OOHDM/) .
- [5] D.SCHWABE, AND G.ROSSI. The object oriented hypermedia design model. *Communications of the ACM* 38, 8 (Aug 1995).
- [6] F.GARZOTTO, D.SCHWABE, AND P.PAOLINI. Hdm: A model based approach to hypermedia application design. *ACM transactions of on information systems* 11, 1 (Jan 1993), 1–26.
- [7] GINIGE, A., B.LOWE, D., AND JOHN ROBERTSON. Hypermedia authoring. *IEEE Multimedia* (winter 1995).
- [8] GOSLING, J., AND GILTON, H. M. Java language environment a white paper. Sun Microsystems computer company, Oct 1995.

- [9] MOTRO, A. Baroque: A browser for relational databases. *ACM transactions on Office Automation Systems* 4, 2 (April 1986), 164–181.
- [10] R.ELMASRI, AND S.NAVATHE. *Fundamentals of Database Systems*, second ed. The Benjamin/Cumings Publishing Company, 1990.
- [11] RUMBAUGH, J., BLAHA, M., PREMIERLANI, W., EDDY, F., AND W.LORENSEN. *Object Oriented Modeling and Design*. Prentice Hall Inc., 1991.
- [12] SCHWABE.D, AND S.D.J.BARBOSA. Navigation modeling in hypermedia applications. Available At: <ftp://ftp.inf.puc-rio.br/pub/docs/techreports/94-42-barbosa.ps.gz>.
- [13] SHNEIDERMAN, B. Designing information abundant websites. Available At: <http://www.cs.umd.edu/users/ben/index.html>, 1997.
- [14] T.ISAKOWITZ, E.A.STOHR, AND P.BALASUBRAMANIYAN. Rmm: A methodology for structured hypermedia design. *Comunications of the ACM* (Aug 1995), 34–48. Also available at: <http://is-2.stern.nyu.edu/tisakowi/ps-files/rmd.zip>.
- [15] WALL, L., AND SCHWARTZ, R. L. *Programming Perl*, first ed. Nutshell Handbooks. O'Reilly and Associates, Inc., 632 Petuluma Avenue, Sebastopol, CA 95472, January 1991. Online reference at: <http://www.perl.com/perl/>.